ServiceCenter®

Version 3

SCAuto for Unicenter AMO

May 2000

Peregrine Systems, Inc.
3611 Valley Centre Drive
San Diego, CA 92130

Peregrine SYSTEMS®
The Infrastructure Management Company™

TOC PREV NEXT INDEX

This edition applies to version 3 of the licensed program of **ServiceCenter**®
and version 1.0 of **SCAuto for Unicenter AMO**

Contents

TOC PREV NEXT INDEX

Preface

Overview

Welcome to Peregrine Systems' *SCAuto for Unicenter AMO*. SCAuto for Unicenter AMO is part of the suite of SCAutomate (SCAuto) interface products that integrate ServiceCenter with premier network and systems management tools.

This guide describes how to implement the SCAutomate interface for integration with Peregrine Systems' ServiceCenter.

Additional information about SCAutomate can be found in the ***SCAutomate Applications for Windows NT and UNIX Guide***.

Prerequisite knowledge

This guide assumes that you have:

- Thorough knowledge of the operation of both ServiceCenter and Unicenter AMO and of the operating system platforms on which those products are installed. This may include Unix and/or OS/390 in the case of ServiceCenter.
- Prior experience customizing ServiceCenter, or equivalent training.
- Prior experience customizing Unicenter AMO, or equivalent training.
- Experience in installation and operation of products on Windows NT.
- Knowledge of Microsoft JScript, Netscape JavaScript, or ECMAscript.

Important: ServiceCenter installation requirements are specific to the machine where ServiceCenter is being installed. These requirements are listed in the respective installation guides.

Contacting Peregrine Systems

Contact one of the Peregrine Systems Customer Support offices listed here if you have questions about, or problems with, ServiceCenter systems.

For more information about Customer Support, check the support web site: **http://support.peregrine.com** Please contact Customer Support for an account on this site.

Note: Only the European Customer Support staff is multilingual and can provide technical support to customers in their native language.

North and South America

To get help immediately, call Peregrine Customer Support at:

(1) (800) 960-9998

For ServiceCenter questions or information, send a fax or e-mail to:

Fax:	**(1) (858) 794-6028**
E-mail:	**support@peregrine.com**

Send materials that Peregrine Systems Customer Support requests to:

Peregrine Systems, Inc.
ATTN: Customer Support
12670 High Bluff Drive
San Diego, CA 92130

Note: Countries outside North and South America are covered by regional offices. Customers should contact the regional office under which their country is listed.

United Kingdom regional office

Great Britain, Greece, and South Africa

Peregrine Systems Ltd.
1st Floor
Ambassador House
Paradise Road
Richmond, Surrey, Great Britain, TW9 1SQ

Phone:	**0800 3345844**
E-mail:	**uksupport@peregrine.com**

France regional office

France, Spain, Italy, Greece, and Africa (except South Africa)

Peregrine Systems
Tour Franklin-La Défense 8
92042 Paris La Défense Cedex, France

Phone:	**+33 (0) (800) 505 100** (International Toll Free)
E-mail:	**frsupport@peregrine.fr**

Germany regional office

Germany and Eastern Europe

Peregrine Systems GmbH
Bürohaus Atricom
Lyoner Strasse 15,
60528 Frankfurt, Germany

Phone:	**0049 6966 80260**
E-mail:	**fweidman@peregrine.com**

Nordic regional office

Denmark, Norway, Sweden, Finland, and Iceland

Peregrine Systems A/S
Naverland 2, 12 SAL
DK-2600 Glostrup
Denmark

Phone:	**+45 80307676**
	+45 43467676
E-mail:	**nordic@peregrine.com**

Benelux regional office

Netherlands, Belgium, and Luxembourg

	Peregrine Systems BV **Botnische Golf 9a** **3446 CN Woerden** **Netherlands**	
Netherlands	Phone:	**+31 3484377070**
Belgium and Luxembourg	Phone:	**0800 74747575**
	E-mail:	**benelux.support@peregrine.com**

Asia-Pacific regional offices

Australia, Hawaii, Hong Kong, Japan, Korea, Malaysia, New Zealand, Singapore

Australia	Phone:	**(800) 146-849**
Hawaii	Phone:	**(1) (800) 960-9998**
Hong Kong	Phone:	**(800) 908056**
Japan	Phone:	**(0044) 221-22795**
Singapore	Phone:	**(800) 1300-949** or **-948**
	E-mail:	**apsupport@peregrine.com**

Chapter 1 Introduction

Overview

SCAuto for Unicenter AMO is a uni-directional gateway between ServiceCenter and CA's Unicenter AMO, providing continuous, automatic forwarding of information from AMO to ServiceCenter ICM on a 24x7 basis. Running as a Windows NT service, it uses state-of-the-art Component Object Model (COM) objects and ActiveX scripting to provide maximum flexibility and extensibility.

Because Unicenter AMO does not provide an API, SCAuto for Unicenter AMO uses Microsoft ADO to query the AMO database. Because Unicenter AMO does not provide a notification mechanism to drive external applications when AMO inventory updates take place, SCAuto for Unicenter AMO periodically selects updated information from the AMO database by selecting units which have been updated since the last poll. This interval is customer settable.

To provide maximum flexibility and ease of customization, the SCAuto for Unicenter AMO product uses state-of-the-art Windows technologies:

- OLE Automation, which defines dynamic COM interfaces to permit scripting languages to use COM objects
- Scripting languages such as VBScript and JScript, which provide rapid application development

SCAuto for Unicenter AMO leverages these technologies to provide a robust, user-tailorable gateway without the inflexibility normally associated with C/C++ applications.

Theory of Operation

SCAuto for Unicenter AMO is designed to propagate information from AMO to ServiceCenter. It works by polling the AMO database for information at a user-defined interval using Microsoft ADO and then converting the information received into ServiceCenter event messages to update the ServiceCenter ICM repository.

SCAuto for Unicenter AMO does not update the AMO database in any way. It requires only read-only access to the database. Both SQL Server and Access are supported as AMO database types.

Delivered Functionality

As delivered, the gateway performs periodic one-way propagation of inventory information from AMO to ICM. While the technology of SCAuto for Unicenter AMO could easily support bi-directional operation, CA does not permit user or third party updates to the AMO database.

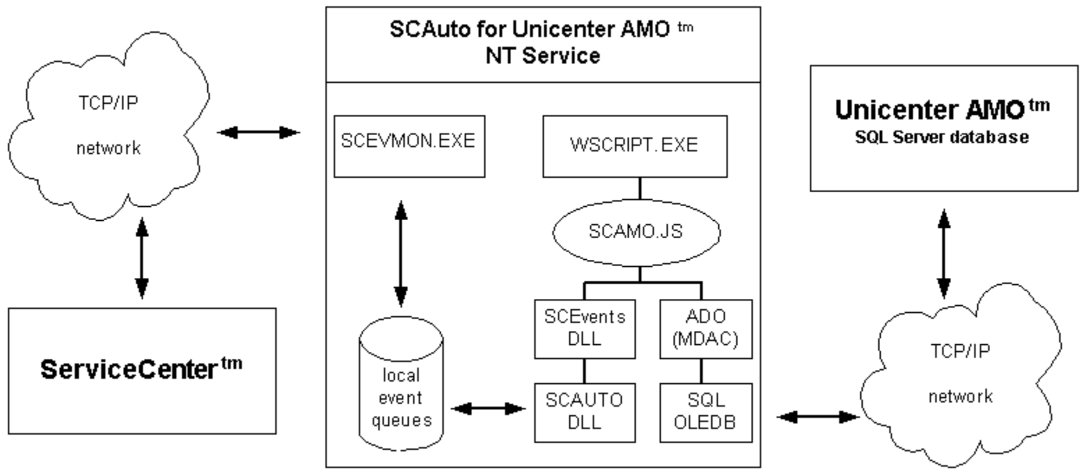
Extensibility

Although SCAuto for Unicenter AMO is delivered with a single script that collects information regarding both hardware and software, the gateway is not limited to this script. The gateway can be customized to run different scripts or additional scripts, simply by changing the *scamo.cfg* file as discussed in Chapter 3.

TOC PREV NEXT INDEX

Architecture

The diagram below illustrates the architecture of SCAuto for Unicenter AMO.



Note the following:

- There are three major components—ServiceCenter, Unicenter AMO, and SCAuto for Unicenter AMO (the gateway between the two).
- Each component can be installed on the same or a different server.
- Communication between the SCAuto for Unicenter AMO gateway and ServiceCenter is based on the SCAuto API, which uses TCP.
- Communication between the SCAuto for Unicenter AMO gateway and the Unicenter AMO database is accomplished using Microsoft ADO, which uses OLEDB or ODBC depending on whether the database is SQL Server or Access.
- ServiceCenter can be running on any supported OS platform (Windows NT, Unix, or MVS).
- For this release of SCAuto AMO, the Unicenter AMO database must either be in SQL Server or be a Microsoft Access .MDB file. Contact Peregrine support for information regarding support of other types of database.
- The SCAuto for Unicenter AMO gateway requires a Windows NT server platform. It does not run on Unix.

Product Components

SCAuto for Unicenter AMO provides a set of OLE Automation objects and redistributes the latest version of the Microsoft Data Access objects, including ADO. These objects are then exploited by some JScript routines (source code provided), which implement a mapping between the AMO database and the ServiceCenter ICM application.

SCAuto for Unicenter AMO has a modular architecture that includes the following building blocks:

- **SCEVENTS.DLL** – an OLE Automation object that encapsulates SCAUTO.DLL, the ServiceCenter API, which uses TCP sockets to communicate with ServiceCenter.
- **SCRUTIL.DLL** – an OLE Automation object which provides utility functions for logging messages and obtaining run-time parameters.
- MAKESCINFO.JS – An ActiveX script written in Jscript which rebuilds the small SQL table called SCINFONAME in the supplied Microsoft Access database called SCAMO.MDB. This table defines how AMO information should be mapped to ServiceCenter Inventory elements. The table consists of 1 row per AMO hardware inventory element. This script need not be executed unless your AMO database contains additional hardware data elements not already listed in the supplied SCINFONAME table.
- **SCAMO.JS** – an ActiveX script written in JScript which propagates hardware and software inventory information from AMO to ServiceCenter, by continuously performing these steps:
    - Select all machines known to AMO which have been updated by AMO since the last poll by SCAMO.JS.
    - Select all hardware inventory items for each machine.
    - Select all software inventory available for each machine.
    - Generate an ICMpc or ICMserver event to ServiceCenter for each machine.
    - Wait for a user-specified time.
- **WSCRIPT.EXE** – The Microsoft Windows Scripting Host.

- **JSCRIPT.DLL** – Microsoft 5.0 ActiveX scripting engine.
- **SCEVMON.EXE** – a daemon which manages communication of ServiceCenter events between SCAuto for Unicenter AMO and ServiceCenter. It provides event message queuing so that ServiceCenter can be started and stopped without affecting the operation of the gateway.
- **SCAMOSRV.EXE** – A Windows NT service to run the previously described components continuously in the background.

Customization

Customization of the SCAuto for Unicenter AMO gateway ordinarily consists of adjusting the mapping of AMO inventory data into specific fields within the Inventory and Configuration (ICM) database in ServiceCenter.

Although it is not necessary to perform any customization prior to starting SCAuto for Unicenter AMO, you may find that some additional customization is necessary in order to populate the tables of the ServiceCenter ICM application to your satisfaction.

Customization is accomplished primarily on the SCAuto AMO side (as opposed to ServiceCenter) by making entries in the SCNAME column of the SCINFONAME table supplied in the SCAMO.MDB database, using Microsoft Access, or a similar tool. Details regarding this process are given in Chapter 5. This type of customization is normally sufficient and does not require any programming skills or ServiceCenter customization training.

Advanced customization can be done by extending the ServiceCenter ICM schema and forms to add additional data fields which are discovered by AMO, but not present in ServiceCenter's ICM tables. These additional fields can then be added to the ICM event maps, whereupon SCAuto AMO can be instructed to map data into them via the process discussed in Chapter 5. This level of customization requires advanced ServiceCenter customization skills. Some general guidelines regarding this type of customization are provided in Chapter 4. However, you will need to consult the appropriate ServiceCenter documentation, or a qualified ServiceCenter consultant.

Finally, it is possible to customize the product by writing new scripts similar to SCAMO.JS, or by modifying the supplied JScript code of SCAMO.JS. This is not normally necessary or recommended. This requires advanced skills in script programming and debugging and should generally only be done by Peregrine personnel or a qualified consultant. Peregrine's ability to support your installation may be limited if you modify any script code on your own.

Chapter 2 Installation

Overview

To operate SCAuto for Unicenter AMO, the following products must be installed:

- ServiceCenter
- Unicenter AMO 3.0
- SCAuto for Unicenter AMO
- Windows Script 5.0
- Microsoft MDAC 2.1 GA SP 2

This document provides installation instructions for only the last three items (SCAuto for Unicenter AMO and the Windows Script 5.0). It is assumed the first two items (ServiceCenter and Unicenter AMO) are already installed on your system. If not, be sure to install them before proceeding with the instructions provided in this document.

Installation checklist

Verify the following items before proceeding:

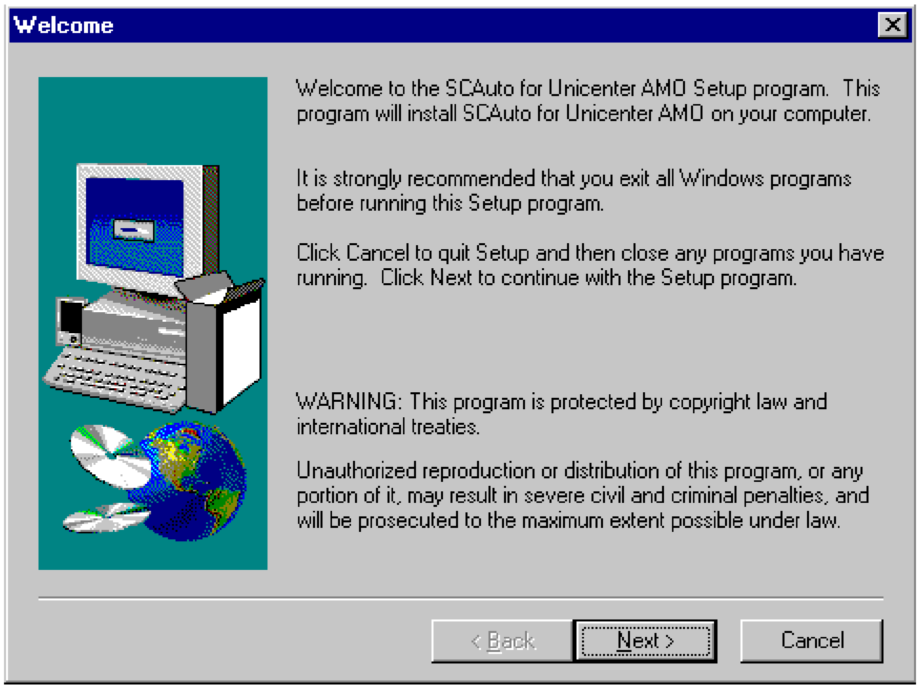
- ServiceCenter 2.1 SP 3E2 or later RTE is installed on your system, and all Event Services applications are at the same level.
- Unicenter AMO 3.0 is installed on your system and you know the name of the SQL Server, the name of the database, and the name and password of a read-only account that can access the AMO database. Alternatively, you can use a Microsoft Access copy of the AMO database. In this case, you must know the path to the .MDB file.
- ServiceCenter is properly configured to run Event Services, including the SCAUTOD server.
- You have collected the information needed to configure the gateway:
  - ServiceCenter hostname.
  - TCP port number of the ServiceCenter SCAUTOD server.
  - AMO database information previously described.

SCAuto for Unicenter AMO

To install SCAuto for Unicenter AMO:

1. Insert the SCAuto for Unicenter AMO CD and run *setup.exe*.

   The Setup program runs until the Welcome screen is displayed.



Welcome to the SCAuto for Unicenter AMO Setup program. This program will install SCAuto for Unicenter AMO on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

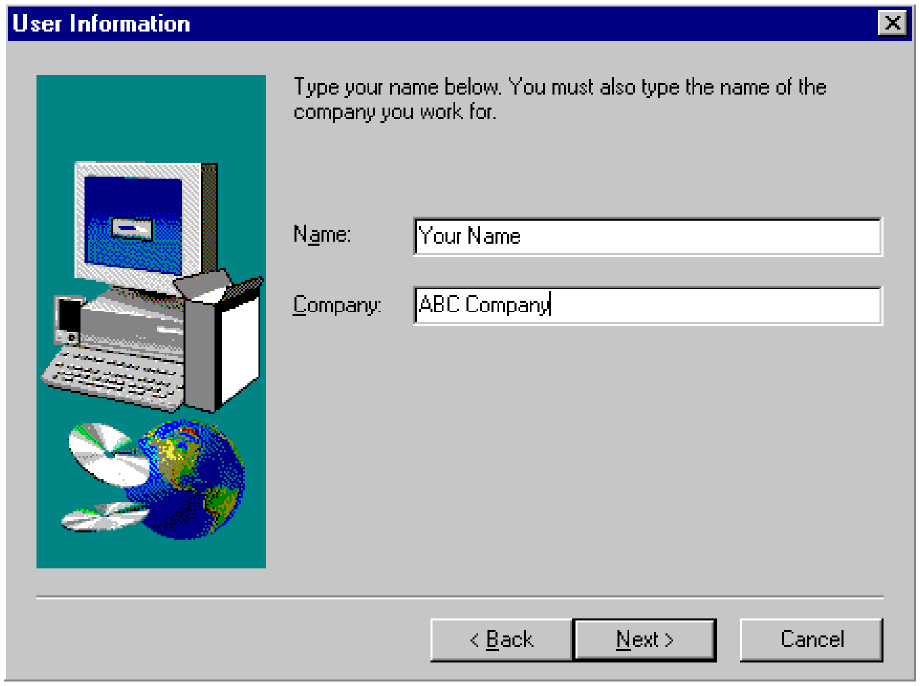
Figure 2-1 Welcome

2. Click **Next** to continue.

   The Readme Information screen is displayed.

3. Click **Next** to continue.

The User Information screen is displayed.


User Information

Type your name below. You must also type the name of the company you work for.

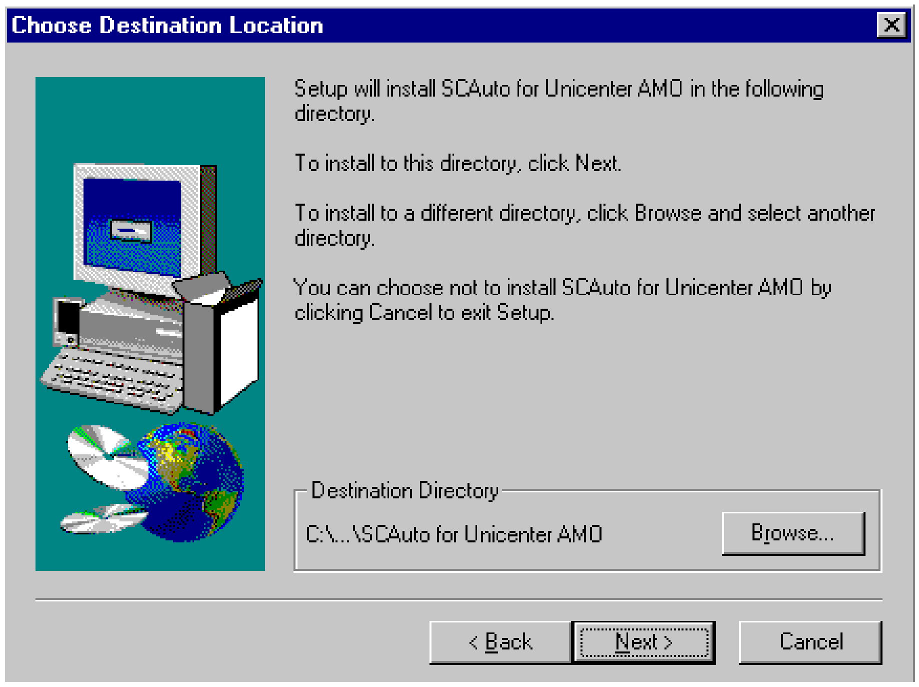
Name: Your Name

Company: ABC Company

Figure 2-2 User Information

4. Type your name and your company name, and then click **Next**.

The Choose Destination Location screen is displayed, which shows the directory to which SCAuto for Unicenter AMO will be installed.



Choose Destination Location

Setup will install SCAuto for Unicenter AMO in the following directory.

To install to this directory, click Next.

To install to a different directory, click Browse and select another directory.

You can choose not to install SCAuto for Unicenter AMO by clicking Cancel to exit Setup.

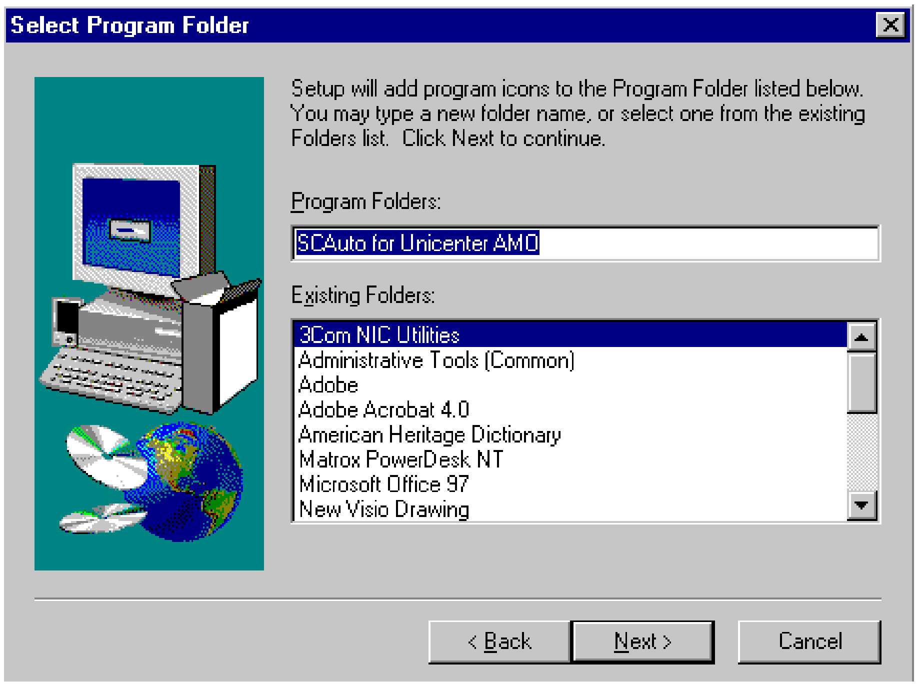
Destination Directory

C:\...\SCAuto for Unicenter AMO

Figure 2-3 Destination Location

5. Ensure that the path shown in the Destination Directory is correct.
   - If so, click **Next**.
   - If not, click **Browse** and select a different directory and then click **Next**.

The Select Program Folder screen is displayed, where you select the program folder to which you want program icons added.



Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

Program Folders:

SCAuto for Unicenter AMO

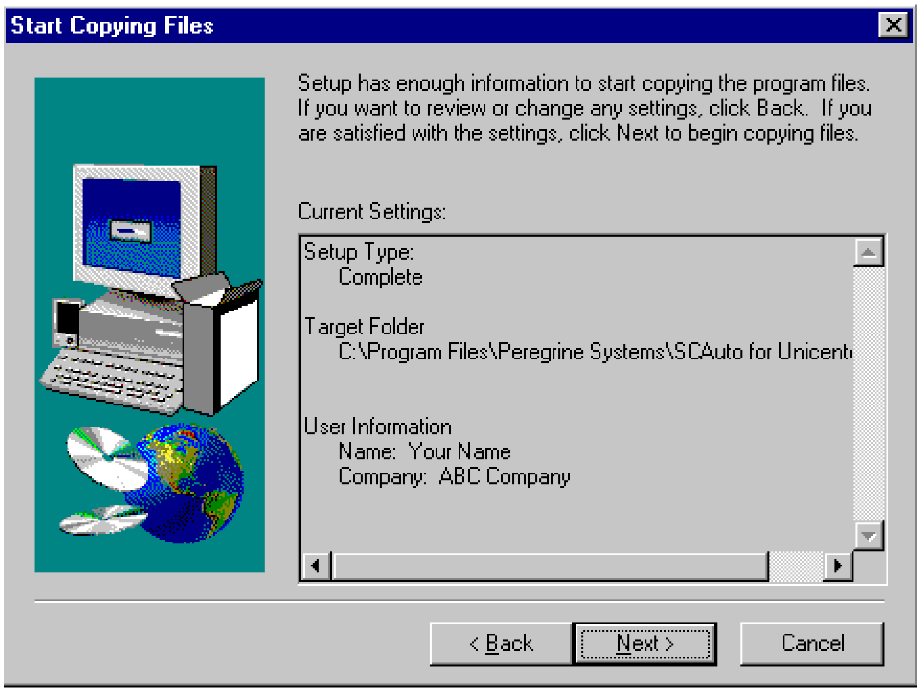
Existing Folders:

3Com NIC Utilities
Administrative Tools (Common)
Adobe
Adobe Acrobat 4.0
American Heritage Dictionary
Matrox PowerDesk NT
Microsoft Office 97
New Visio Drawing

Figure 2-4 Select Program Folder

6. Select a program folder for your SCAuto for Unicenter AMO system, and then click **Next**.

The Start Copying Files screen is displayed, indicating that Setup is ready to start copying the SCAuto for Unicenter AMO program files.



Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

Setup Type:
    Complete

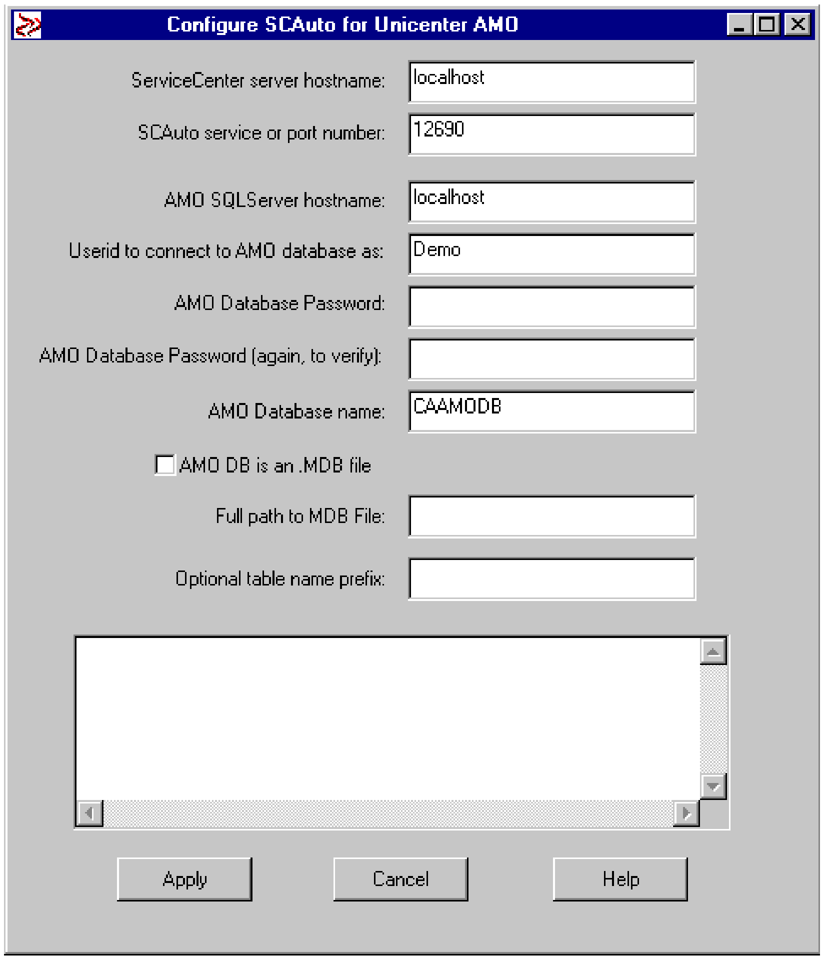
Target Folder
    C:\Program Files\Peregrine Systems\SCAuto for Unicent

User Information
    Name: Your Name
    Company: ABC Company

Figure 2-5 Start Copying Files

7. After verifying that the current settings displayed are correct, click **Next** to start copying the files.

After all files are installed on your system, a dialog is displayed allowing you to edit your *scamo.ini* file settings. Refer to the provided help or the section on the *scamo.ini* file in *Chapter 3* of this guide for further information.


Configure SCAuto for Unicenter AMO

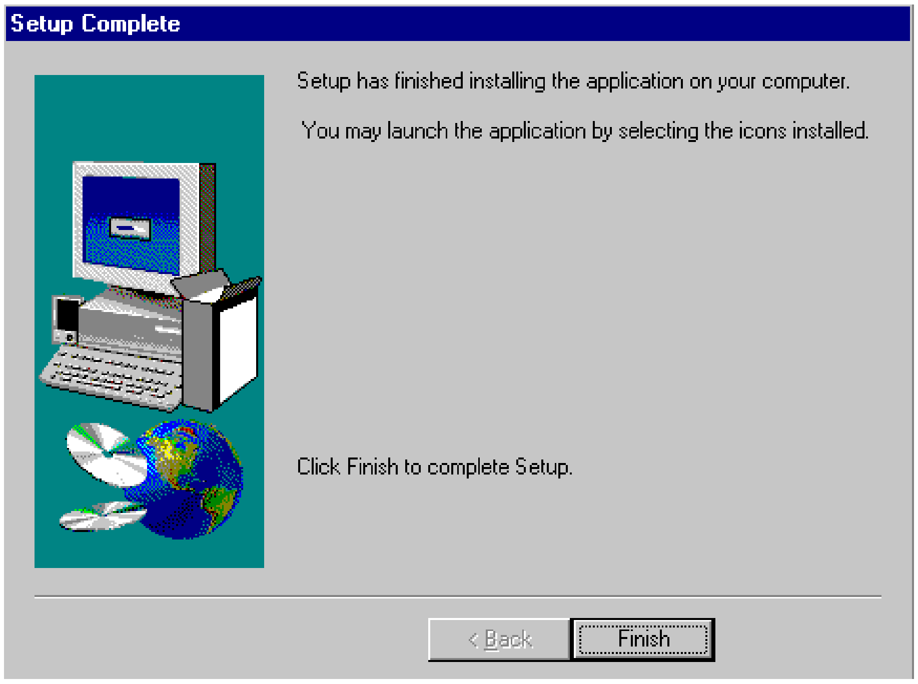
ServiceCenter server hostname:	localhost
SCAuto service or port number:	12690
AMO SQLServer hostname:	localhost
Userid to connect to AMO database as:	Demo
AMO Database Password:	
AMO Database Password (again, to verify):	
AMO Database name:	CAAMODB
☐ AMO DB is an .MDB file	
Full path to MDB File:	
Optional table name prefix:	

Figure 2-6 Configuration Screen

8. Ensure that all settings are correct and then click **Apply**.

Note: These settings can be changed later by choosing **Programs>SCAuto for Unicenter AMO>Configure SCAuto for Unicenter AMO** from your Windows Start menu.

The Setup Complete screen is displayed when Setup has finished installing SCAuto for Unicenter AMO on your computer.



Setup has finished installing the application on your computer.

You may launch the application by selecting the icons installed.

Click Finish to complete Setup.

Figure 2-7 Setup Complete

9. Click **Finish** to exit.

   SCAuto for Unicenter AMO is now installed.

10. After finishing the installation and configuration of all components, refer to *Chapter 7* of this guide for further information on operating your SCAuto for Unicenter AMO system.

Note: If you already have your own custom script, you will want to copy this script into your installation\bin directory.

Windows Script 5.0

SCAuto for Unicenter AMO requires Windows Script 5.0, which includes Windows Scripting Host and the 5.0 scripting engines, including JScript 5.0.

At the conclusion of the setup procedure described above, the Microsoft setup procedure for Windows Script 5.0 (US English version) will be automatically launched. If you wish to obtain and install a more current version or a version for a different locale, cancel this part of the setup and do the following:

1. Access the web site http://www.microsoft.com/msdownload/vbscript/scripting.asp.
2. Despite the *vbscript* in the URL, you are taken to the Windows Script page, where you can download **ste50en.exe** (900K), which is the US English version, or an equivalent localized version, such as **ste50de.exe**, or **ste50fr.exe**. The download includes Windows Script 1.0, JScript 5.0, VBScript 5.0, and other required files.
3. Download the appropriate Windows Script file and install it. You must be running version 5.0 or later of the Windows Scripting engines to successfully run the Microsoft JScript code provided with SCAuto for Unicenter AMO.
4. You may wish to download the free script debugger, unless you have Visual Studio Professional or Enterprise Edition, which includes an enhanced debugger.
5. We also recommend you download the Microsoft documentation package for JScript 5.0.

TOC PREV NEXT INDEX

Microsoft MDAC (ADO)

SCAuto for Unicenter AMO also requires MDAC 2.1 GA SP 2 or later.

At the conclusion of the Windows Script part of the setup, the Microsoft setup procedure for MDAC 2.1 GA SP 2 is automatically launched.

Chapter 3 Configuration

Overview

To configure SCAuto for Unicenter AMO, you may need to make changes to the following files:

scamo.ini

Changes to this file are rarely needed once a few key parameters are correctly set. These include the identities of the ServiceCenter and Unicenter AMO servers, and the User ID and password required for connection to the Unicenter AMO database. The latter parameters should have been correctly set when you ran the configuration dialog at the end of the installation process. If you need to change those, it is recommended you rerun the configuration dialog rather than hand-edit the scamo.ini file. However, you may need to modify the scamo.ini file by hand for certain things, such as activating debug logging. A discussion of the required scamo.ini file parameters follows. A complete list of .ini parameters is also given in Appendix A.

scamo.cfg

Changes to this file should not be necessary. An exception might be the specification of some transient NT command you wish executed at service startup or the specification of additional scripts to be run as part of the gateway.

scamo.ini File

The following parameters in the scamo.ini file may need to be configured:

sessid:SC_AMO

Unique identifier given to this instance of SCAuto for Unicenter AMO.

sceventserver:localhost.12690

The *sceventserver* parameter defines the TCP/IP hostname and TCP port number of the ServiceCenter SCAUTOD server. The two values must be specified together, separated by a period. The SCAUTOD TCP port number must not be confused with the other two TCP port numbers used by ServiceCenter (express client server and full client server). The value *localhost* in this example indicates that the ServiceCenter server is running on the same machine as the gateway. The *12690* value is the default TCP port used by the SCAUTOD server in ServiceCenter.

stopeventname:SCAutoAMO.StopEvent

The *stopeventname* parameter specifies the name of the Win32 event to be used to synchronize SCAuto AMO processes.

noeventsfromsc:1

The *noeventsfromsc* parameter, not implemented in version 1.0, specifies whether to ignore events sent by ServiceCenter. When scripts are provided to implement a bi-directional dialog between Unicenter AMO and ServiceCenter, set this parameter to 0 to enable the dialog.

sleep:3600

The *sleep* parameter specifies the interval in seconds between Unicenter AMO database polls.

retrymax:3

The *retrymax* parameter specifies the number of times the SCAuto AMO service will attempt to restart a failing process before giving up and shutting down the service.

lastrundate:0

The lastrundate parameter is initially set to zero, as shown above. After the first execution of SCAuto AMO, this parameter is used to store the largest AMO LRUNDATE value observed during the each poll of the AMO database. Each time SCAuto AMO wakes up and polls the AMO database, it selects machines from the AMO unit table whose LRUNDATE value is greater than this value. If this value is manually set back to zero, SCAuto for AMO will process all AMO machines, regardless of the values of LRUNDATE in the AMO UNIT table.

amo_dbtype:

The *amo_dbtype* parameter defines what kind of database is in use by Unicenter AMO. A type of MDB indicates an Access database. SQLOLEDB indicates SQL server.

amo_sqlserver:

The *amo_sqlserver* parameter defines the TCP/IP hostname of the Unicenter AMO SQL server.

amo_username:

The *amo_username* parameter defines the User ID for SCAuto to use when logging in to Unicenter AMO when SQL Server is the database.

amo_password:

The *amo_password* parameter defines the SQL Server password for SCAuto to use when logging in to Unicenter AMO. This value is stored in encrypted form and should not be edited by hand.

amo_dbname:

The *amo_dbname* parameter defines the name of the Unicenter AMO database to be used for updating the ServiceCenter database. This is a required parameter.

amo_dbpath:

The *amo_dbpath* parameter defines the path to the Unicenter AMO database defined in the *amo_dbname* parameter. This parameter is only valid if the AMO database is a Microsoft Access database.

amo_dbprefix:

The *amo_dbprefix* parameter describes a standard prefix that is applied to all tables in the Unicenter AMO database. This parameter is supplied as a convenience for using SCAuto AMO with an AMO database in Microsoft Access (.mdb) format, where the .mdb file was created by importing a SQL Server database into Access. When this is done, the name of each AMO table is typically prefixed with the name of the table owner, for example, **dbo_**. The default value for this parameter is **dbo_**. This parameter is only valid if the AMO database is a Microsoft Access database.

domainid:

The *domainid* parameter defines whether or not the SCINFONAME table must consider all permutations of ITRTNMID, ITPRNMID, and ITNAMEID values when it is built. If domainid is set to 0 (recommended), ignore domainid when building the SCINFONAME table and performing mapping. If domainid is set to 1 (not recommended), you must rebuild SCINFONAME by running makescinfo.js which will define the mapping for each combination of DOMAINID, ITRTNMID, ITPRNMID and ITNAMEID value.

SCAutomate for Unicenter AMO as delivered assumes that ITRTNMID, ITPRNMID, and ITNAMEID values are the same from domain to domain. This recommended setting limits the size of the SCINFONAME table, making it easier to define field mapping to ServiceCenter inventory. This might not be the case, for example, when language requirements dictate different names for the same unit. Note that setting the domainid to 1 greatly increases the size of the SCINFONAME table, making ServiceCenter field mapping a difficult task.

unittypes:

The *unittypes* parameter defines which AMO Unit Types should be collected for insertion in the ServiceCenter inventory. If this parameter is provided, all unit types other than those specified will be excluded. If the parameter is not supplied, all unit types will be included. The unit types must be separated by commas and enclosed in parentheses. For example:

unittypes:(1,100,120)

excludeitems:

The *excludeitems* parameter defines which items (the ITRTNMID value) should be excluded when building the SCINFONAME table. This parameter is used to limit the size of the SCINFONAME table, since many of the items in the AMO database are not candidates for mapping to ServiceCenter inventory. The items must be separated by commas and enclosed in parentheses, e.g.:

excludeitems:(2310,2318)

Important: The following parameters in the scamo.ini file do not need to be modified. These should only be changed at the direction of Peregrine support personnel.

log:scamo.log

Indicates that the name of the message log file is to be "scamo.log".

debugscautoevents:0

If 1, additional messages are logged in the indicated log file.

debugprocesses:0

If 1, additional messages are logged in the indicated log file.

event_map_dir:EventMap

Specifies the directory (relative to installation directory) where event maps are kept.

scamo.cfg File

The scamo.cfg file is read by the SCAuto for Unicenter AMO service manager program (SCAMOSRV.EXE) at startup, and controls the commands and processes executed by the service. Each line in the file must be a valid Windows NT command.

There are two types of commands that can be specified:

- Transient NT commands, such as might be issued from an NT command line. These must be prefixed with *CMD /c*. Once execution of the command finishes, the command terminates. These can be used to perform maintenance tasks (such as moving or renaming a log file) each time the service is started.
- Long-running commands, which execute until the service is shut down. The long-running commands relevant to SCAuto for Unicenter AMO are discussed in the following paragraphs.

Transient NT commands

As distributed, no transient NT commands are specified in the scamo.cfg file.

Long-running commands

The following long-running commands are relevant to SCAuto for Unicenter AMO:

SCEVMON.EXE

scevmon.exe is a background process that is responsible for communication with ServiceCenter. It runs until the service is shut down. It makes a TCP connection with the SCAUTOD server component of ServiceCenter, using the hostname and port information supplied via the **sceventserver** parameter of the *scamo.ini* file. There should be no need to modify this line of scamo.cfg. If you do, SCAuto for Unicenter AMO will probably not operate correctly.

WSCRIPT.EXE

This command invokes the Microsoft scripting host to execute the scamo.js, scamosw.js and makescinfo.js scripts. The JScript code in scamo.js and scamosw.js implements the actual logic of the gateway, and makescinfo.js builds the mapping table.

Chapter 4 Customizing ServiceCenter

Overview

There is no need to customize ServiceCenter in order to obtain immediate benefits from SCAuto for Unicenter AMO, other the possible installation of any *unload* files you may need to apply. Check the README.TXT file supplied on the CD to determine if any such maintenance is required.

However, some customers may eventually wish to modify the ServiceCenter ICM application to contain additional data items which are collected by AMO, but are not part of the *out-of-the-box* ICM schema. In order not to duplicate a lot of ServiceCenter documentation, a complete discussion of this topic is not given here, but a general description of the required steps is given. Before attempting this procedure, be sure that:

- Both ServiceCenter and the SCAuto for Unicenter AMO product are operating normally.
- You have the appropriate ServiceCenter documentation on hand, including the ServiceCenter Application Administration, Data Administration, System Tailoring, and Event Services guides.
- You have the prerequisite ServiceCenter training and/or experience.

Extending the ICM Schema to Accomodate Additional AMO Data

To extend the ICM schema to accomodate additional AMO data.

1. Back up the ServiceCenter database before making any changes.
2. Use the Database Dictionary tool to add the required fields to the appropriate ICM tables. Fields which are not device-specific should go in the *device* table, and device-specific fields should go in the particular *attribute* table, such as the *pc* table or *server* table.
3. Use the Forms Designer tool to add the new fields to the appropriate form, such as **device.pc.g**.
4. Rebuild the automatically generated event maps for the ICM device type in question. For example, if you are added fields to the *pc* file, you must rebuld the *ICMpc* input event definition. Some versions of ServiceCenter have an application called *Build new ICM Event Map* which do this for you automatically. For older versions of ServiceCenter, you may have to manually make the changes to the event definition. The objective is to add the new fields to both the database and the event map in Event Services.
5. It is necessary to inform the SCAuto for Unicenter AMO service that the event map has been changed. SCAMO caches the event maps it uses in the \EventMap subdirectory of the SCAuto for Unicenter AMO directory. Stop the SCAuto for AMO service, then navigate to the \EventMap directory and delete any .input file which has the same name as the event map which you updated in ServiceCenter.
6. Make sure ServiceCenter is available, then restart the SCAuto for AMO service. While SCAuto AMO does not normally need ServiceCenter to be available when it runs, it must have ServiceCenter available the first time it is started after deleting any *.input* file, so that it can reconstruct the missing .input file by querying ServiceCenter about the updated event map.

Chapter 5 Customizing SCAuto for Unicenter AMO

Overview

Because AMO and ServiceCenter ICM have different schemas for hardware inventory, a process called *mapping* is used to define where the various pieces of AMO hardware data are to be stored in the ServiceCenter inventory files.

SCAuto for Unicenter AMO is delivered with default a set of *mapping instructions* for mapping AMO hardware into ServiceCenter's inventory files.

The mapping for hardware inventory is discussed in this chapter, and can be modified by the customer.

Software Inventory Mapping

Software inventory is handled differently than hardware inventory. Because software is simply stored in ServiceCenter as a small set of arrays of string data for each PC, mapping for software inventory is expressed directly in the JScript code in SCAMO.JS. No external table mechanism is presently provided to control how or where the software data is stored.

The relevant arrays in ServiceCenter ICM database are **sw.name** and **sw.vendor**. These arrays are displayed when you click on the "Software" tab in the ICM application for a PC, server, or workstation. The AMO software-related fields which are currently extracted from the AMO database are:

- APTITLE – title of the software application, including version, i.e. Microsoft Word 97
- PATH – directory containing the executable, i.e. C:\PROGRAM FILES\MICROSOFT OFFICE\OFFICE
- APFNAME – file name of the executable, i.e., WINWORD.EXE
- PUBNAME – the name of the software publisher, i.e. Microsoft Corporation

The first three of these items are concatenated together and appended to the **sw.name** array. The publisher name is appended to the **sw.vendor** array.

The SCINFONAME Table

The SCINFONAME table is contained in a small Microsoft Access database that is supplied with SCAuto for Unicenter AMO as an *.mdb* file. It contains the mapping instructions that relate AMO information to corresponding attributes in ServiceCenter's inventory repository. *Mapping instructions* is the term given to the entries in the SCNAME column. The table below shows only the default mapping instructions.

Note: The DOMAINID for all entries in this table is 0.

UNITTYPE	ITRTNMID	ITPRNMID	ITNAMEID	ITNAME1	ITNAME2	ITNAME3	SCNAME
1	5	2	186	$GeneralInventory$	$Display Adapter$	Type	video
1	5	2	144	$GeneralInventory$	$Display Adapter$	Resolution	video
1	5	2	104	$GeneralInventory$	$Display Adapter$	Name	video
1	5	2	18	$GeneralInventory$	$Display Adapter$	Color depth	video
1	5	2	9	$GeneralInventory$	$Display Adapter$	BIOS info	video
1	5	2	190	$GeneralInventory$	$Display Adapter$	Vendor	video
100	2309	94	192	NetWare Server Basic Information	Public Information	Version	version
100	2309	94	339	NetWare Server Basic Information	Public Information	Revision	version
120	2303	2304	192	NT Server inventory	General information	Version	version
100	2309	94	190	NetWare Server Basic Information	Public Information	Vendor	vendor
1	56	77	189	Network	Windows Network Information	User Name	user.id
120	2303	2304	542	NT Server inventory	General information	Users	user.id
1	56	91	157	Network	TCP/IP Information	Subnet Mask	subnet.mask
1	5	10	76	$GeneralInventory$	$Network Adapter$	IP Subnet Mask	subnet.mask
1	87	87	2063	User Information	User Information	User Phone Number	primary.phone
1	87	87	2185	User Information	User Information	Phone (888-555-5555)	primary.phone
120	2303	14	127	NT Server inventory	$Stamp$	Platform	operating.system
1	5	10	190	$GeneralInventory$	$Network Adapter$	Vendor	nic
1	5	10	104	$GeneralInventory$	$Network Adapter$	Name	nic
1	5	10	98	$GeneralInventory$	$Network Adapter$	Media	nic
1	56	91	67	Network	TCP/IP Information	Hostname	network.name
120	2303	2304	104	NT Server inventory	General information	Name	network.name
1	5	10	75	$GeneralInventory$	$Network Adapter$	IP Hostname	network.name
1	56	91	72	Network	TCP/IP Information	IP Address	network.address
1	5	10	72	$GeneralInventory$	$Network Adapter$	IP Address	network.address
1	5	2001	104	$GeneralInventory$	$Monitor$	Name	monitor
1	5	2001	190	$GeneralInventory$	$Monitor$	Vendor	monitor
1	87	87	136	User Information	User Information	Product Version	model
1	87	87	135	User Information	User Information	Product Name	model
120	2303	2304	186	NT Server inventory	General information	Type	model
1	2035	87	135	Generic Computer inventory module	User Information	Product Name	model
1	5	10	91	$GeneralInventory$	$Network Adapter$	MAC Address	mac.address
1	5	11	606	$GeneralInventory$	$Network$	Computer Name	logical.name
1	5	11	23	$GeneralInventory$	$Network$	Computer Name	logical.name
100	2309	94	338	NetWare Server Basic Information	Public Information	Server Name	logical.name
100	2309	94	340	NetWare Server Basic Information	Public Information	Revision Date	last.maintenance
1	5	10	234	$GeneralInventory$	$Network Adapter$	Default Gateway	gateway
1	5	4	151	$GeneralInventory$	$Drives$	Size	feature.size
1	5	4	104	$GeneralInventory$	$Drives$	Name	feature.id
1	5	4	195	$GeneralInventory$	$Drives$	Volume Label	feature.description
1	5	10	74	$GeneralInventory$	$Network Adapter$	IP Domain	domain
1	2003	2023	292	Hardware	Processor Information	Main Processor Type	cpu.type
1	2003	2023	293	Hardware	Processor Information	Main Processor Vendor	cpu.type
1	2003	2023	289	Hardware	Processor Information	Main Processor Name	cpu.type
1	5	13	186	$GeneralInventory$	$Processors$	Type	cpu.type
1	87	87	355	User Information	User Information	Full Name	contact.name
120	2303	2304	2780	NT Server inventory	General information	Comment	comments
1	5	15	34	$GeneralInventory$	$System BIOS Information$	Date	bios
1	5	15	68	$GeneralInventory$	$System BIOS Information$	ID String	bios

Note: Entries of a particular ITRTNMID number can be excluded using the **excludeitems** parameter in the *sc.ini* file.

TOC PREV NEXT INDEX

Customizing the SCUNITTYPES Table

The SCUNITTYPES table is contained in a small Microsoft Access database that is supplied with SCAuto for Unicenter AMO as an *.mdb* file. It contains the mapping instructions that relate AMO unit types to corresponding device types in ServiceCenter's inventory repository. The sample below shows only the default device type assignments:

DOMAINID	UNITTYPE	SCDEVICETYPE
0	0	
0	1	pc
0	2	
0	3	
0	100	server
0	110	server
0	111	
0	120	server

Note: Entries of a particular UNITTYPE can be excluded by omitting them from the **unittypes** parameter in the *sc.ini* file. The SCINFONAME table is created based upon those parameters. The DOMAINID value is set dependent upon the **domainid** parameter in the sc.ini file.

You can instruct SCAuto for Unicenter AMO to write all data for units of type 1, for example, to the *server* table rather than to the *pc* table by replacing *pc* with *server.*

Mapping instructions

All AMO data is mapped to ServiceCenter inventory tables by coding what we refer to as *mapping instructions* in the SCNAME column of the SCINFONAME table previously shown. Each row of the SCINFONAME table represents a specific item of AMO hardware data that may be collected for a given PC or server. The textual description of each item (from the AMO INFONAME and INFOTRNM tables) is reproduced in each row. Each row has a column called SCNAME, which, if not blank, specifies the ServiceCenter ICM field name or names which are to receive the data.

Mapping instructions can be as simple as a single ServiceCenter inventory field name (column name) or as complex as a series of JScript expressions separated by semicolons. Both types are discussed in this document.

Simple mapping instructions

The simplest kind of mapping instruction consists of just the name of a scalar (non-array) ServiceCenter inventory field, taken from either the *device* table, the *pc* table, or the *server* table, coded in the SCNAME column of the SCINFONAME table. All of the mapping instructions shipped with the supplied SCINFONAME table are of this type. For example, the first few rows of the SCINFONAME table shown above all contain the value *video* in the last column. Whenever any of these AMO data items are encountered, they are mapped to the ServiceCenter ICM field called *video* of the ICM file called *pc*. Because more than one AMO data item is being mapped to the same ICM field, the AMO data values are concatenated together with a separating blank. Give some thought before mapping multiple AMO fields to a single ICM field. For example, you would not want to map multiple AMO data items containing the same IP address to the ICM field **network.address**, because you would end up with several instances of the PC's IP address concatenated together in that field in ICM, which looks ugly. On the other hand, AMO collects half a dozen or more individual items of video card information, and it is reasonable to concatenate all these together into the ICM field called *video*.

All AMO data for a given machine is directed into the ServiceCenter *device* table and one of the ServiceCenter device type tables, usually *pc* or *server*. Which device type table is updated is controlled by the SCDEVICETYPE column of the row of the SCUNITTYPE table which is selected for a given AMO unit.

For example, if the AMO unit has a UNITTYPE of 100, since the SCDEVICETYPE column for the row for unittype 100 contains *server*, the corresponding ServiceCenter ICM device type will be *server*. Therefore, SCAuto AMO will generate an *ICMserver* event for this AMO unit. The ServiceCenter ICMserver event updates the *device* and *server* tables in ServiceCenter, as defined by the Event Registration settings for the ICMserver event in ServiceCenter Event Services.

If for some reason you wanted all AMO data for both servers and PCs to go to ServiceCenter as *pc* device types, you would simply update the SCUNITTYPE table to change all values of SCDEVICETYPE containing *server* to *pc*.

TOC PREV NEXT INDEX

Mapping into ServiceCenter array fields

AMO data can be mapped into ServiceCenter array fields in exactly the same way as for scalar fields, by simply specifying the name of the array. For example, the *feature.description* field in the ICM *pc* file is an array. When you code an array name in the SCNAME column, the AMO data is placed into the next available position at the end of the named array. Be very careful when mapping data into the *feature.id*, *feature.size*, and *feature.description* arrays. These 3 arrays are displayed as parallel columns in the ICM application, and the element at each offset in any one of the arrays is assumed to relate to the corresponding elements in the other two arrays. So you should never map an AMO data item to one of these arrays unless you also map two related AMO data items to the other two arrays. This can be tricky to do. If you fail to do this, the columns won't line up properly when viewed in ICM. It may occasionally occur that you can identify two related AMO data items which you would like to map to 2 of the feature arrays, but cannot identify an AMO item to map to the third feature array, or the third item is not always available for some reason. To deal with this, you can use the assignment technique (discussed in the section *Advanced Mapping Instructions* that follow) to assign a literal value to the third feature array. For example, assume you have identified an AMO data item which you wish to map to the next row of *feature.id*, and you have further identified an AMO data item which you wish to map to the corresponding row of *feature.desc*, but you cannot identify any related AMO data item to map to the corresponding row of *feature.size*. To deal with this situation, assign a blank value to feature.size as an additional mapping instruction for *one* of the other two items. For example:

```
feature.id; feature.size = ""
```

Another way to map into array fields involves hard-coding the array offset in brackets following the array name. For example:

```
feature.size [ 12 ]
```

will place the current AMO data item in the 12th position in the *feature.size* array. You cannot use a variable for the array offset. It must be a literal number between 1 and n where n is a reasonable value. No specific upper limit exists for n, but the total amount of event data that is permitted for any single ICM event is approximately 32K.

This method allows you to force data to a particular element of an array, but keep in mind that if more than one AMO data item maps to this row of SCINFONAME, the second and subsequent items will overwrite the previous ones at the array position you specified. Therefore, hard-coding a specific array offset is not normally recommended. This feature should only be used if you have a strict requirement to map a predictable number of specific AMO data items to specific rows of the ICM features array.

TOC PREV NEXT INDEX

Advanced mapping instructions

For more advanced applications, a mapping instruction can consist of an assignment statement. Whether the event field is a scalar or an array, it permits you to assign literal data to some other field at the time an AMO data item is mapped to a particular field. The legal values for the right hand side of the assignment statement are:

- Any literal value, either string or number:

SCNAME
user.id = “Anonymous “
contract.id = 0

- Any of the predefined variables ITNAME1, ITNAME2, or ITNAME3 that contain the values taken from the SCINFONAME row being processed:

SCNAME
parent = ITNAME3
network.name = ITNAME2

- The predefined variable *itval* which contains the current AMO data item. Note that coding "user.id = itval" and simply coding "user.id" all by itself are equivalent.

ITNAME3	SCNAME
User Name	comments[1] = “Although user is Anonymous, the contact name is “+itval

- A valid JScript expression involving any combination of literal values, itval, ITNAME1, ITNAME2, ITNAME3 and valid JScript operators or methods

ITNAME3	SCNAME
User Name	comments[1] = “Although user is Anonymous, the contact name is “ + itval + “ or “ + ITNAME3

You may not use an event field specification on the right hand side at this time. In other words,

```
feature.id[2] = feature.id[1] + 10
```

is invalid.

Multiple mapping instructions are allowed, separated by semicolons. This allows you to map the AMO data item to multiple fields, or, when used in conjunction with assignment statements, to assign either a literal or an expression to some other field. For example:

SCNAME
feature.size[1] ; feature.id[1] = ITNAME2 + " " + ITNAME3 ; feature.desc[1] = "Whatever you like"

This would put the current AMO data item into the first position of the *feature.size* array. It will then concatenate the ITNAME2 and ITNAME3 columns from the SCINFONAME row corresponding to that AMO data item with an intervening blank and put that into the first position of the *feature.id* array. Finally, it will put the literal value *Whatever you like* into the first position of the *feature.desc* array. You are not limited by the number of statements separated by semicolons except by the length of the SCNAME column, which is limited by Access to 255 characters.

Note: You do not have to use the same index values in each array assignment, although doing so is advisable when dealing with the features arrays, since this helps keep them lined up.

Debugging

Should you encounter problems in your Mapping Instructions, check the following:

- If you obtain unexpected results, always first examine the *scamo.log* file contained in the SCAuto for Unicenter AMO installation directory. There is usually an error message that indicates what is wrong. If you contact Peregrine support, email this entire log file to them. The most common problems are errors in the data specified in the configuration dialog, which results in SCAuto for Unicenter AMO being unable to connect to the AMO database.
- Make sure that SCAuto AMO was able to start up normally. A normal startup sequence of messages in *scamo.log* looks like this:

```
09/08/1999 12:00:06   pid (402)(326) =================== SCAuto for
Unicenter AMO started ===================
09/08/1999 12:00:06   pid (402)(326) scamosrv.exe (Linked: Fri Aug 13
14:52:25 1999) started
09/08/1999 12:00:06   pid (402)(326) scamosrv.exe: Created stop event
SCAutoAMO.StopEvent
09/08/1999 12:00:06   pid (402)(326) Changing desktop for process
wscript.exe from Default to winsta0\default
09/08/1999 12:00:06   pid (461)(473) scevmon.exe (Linked: Thu Jul 01
12:04:57 1999) started
09/08/1999 12:00:06   pid (461)(473) scevmon: Using stop event
SCAutoAMO.StopEvent
09/08/1999 12:00:06   pid (461)(473) scevmon: "noeventsfromsc" parm
recognized. No events will be retrieved from ServiceCenter EventOut
queue
09/08/1999 12:00:07   pid (461)(473) scevmon : Connected to
localhost.12690
09/08/1999 12:00:10   pid (439)(420) =================== SCAuto for
Unicenter AMO: Script scamo.js started ===================
09/08/1999 12:00:10   pid (439)(420) Home directory is
D:\PROGRA~1\PEREGR~1\SCAUTO~4
09/08/1999 12:00:10   pid (439)(420) Using .ini file
D:\PROGRA~1\PEREGR~1\SCAUTO~4\scamo.ini
09/08/1999 12:00:10   pid (439)(420) Successfully created the SCEvent
object
```

```
09/08/1999 12:00:10   pid (439)(420) Using AMO database connection
string Driver={Microsoft Access Driver (*.mdb)}; DBQ=d:\avnet\db1.mdb;
DefaultDir=d:\avnet\
09/08/1999 12:00:10   pid (439)(420) Using SC Mapping Database
connection string Driver={Microsoft Access Driver (*.mdb)};
DBQ=SCAMO.MDB; DefaultDir=D:\PROGRA~1\PEREGR~1\SCAUTO~4\Data
09/08/1999 12:00:10   pid (439)(420) Successfully created the ADO
database objects
09/08/1999 12:00:16   pid (439)(420) Successfully connected to the AMO
database
09/08/1999 12:00:16  pid (439)(420) Successfully connected to the SCAMO
mapping database
```

If SCAuto AMO was unable to start successfully and you are unable to resolve the problem from the error messages given, contact Peregrine customer support.

- Make sure SCAuto AMO is able to successfully generate ICM events to ServiceCenter. If it is, you should see a series of messages like this in the *scamo.log* file:

```
09/08/1999 12:00:29   pid (439)(420) Successfully logged event
U004WKNT0008^^^555-111-
2222^^^^^^^^^pc^^^^^^^^^^^A:|C:|D:|F:|H:|I:|J:|L:|O:|Y:|Z:|^N/A| |
|SYS|VOL3|DISK2|VOL2|DISK3|DISK2|VOL2|SYS|^^^^172.16.23.201^^bignet.com
^^^07/31/98 Phoenix ROM BIOS ....
```

If there is difficulty generating events, there will be some sort of relevant error messages. If you are unable to resolve the problem from the error messages given, contact Peregrine customer support.

- If events are not being generated, but no error messages appear, either there are no AMO units eligible for selection based on LRUNDATE or they are not being mapped properly. Since SCAuto AMO starts by selecting all AMO units with LRUNDATE greater than zero, mapping problems are the most likely problem.
- Make sure your SCUNITYPE table is mapping the AMO unit type codes to valid ServiceCenter ICM device types. Most of the machines collected by AMO will be of type *1* which should ordinarily be mapped to the ICM *pc* device type. Most of the rest will be of type *100* or *120* which are NT or Netware servers. Other AMO unit types should ordinarily be ignored by omitting any value in the SCDEVICETYPE column. While it is permissible to change the values supplied by Peregrine to another valid value, for example, changing *pc* to *workstation*, be sure the new value you specify corresponds to a valid ServiceCenter ICM device type. Be careful not to accidentally uppercase these values. ServiceCenter is case-sensitive. And if you change any SCUNITTYPE mappings, be aware that this may require changes to the SCNAME column values in the SCINFONAME table, because not all ICM device types have the same named fields.

- Make sure you did not misspell any ICM field names coded in the SCNAME column in the SCINFONAME table. These are case-sensitive. Make sure they match exactly the names in the ServiceCenter Database Dictionary for the particular device type.
- Make sure any advanced mapping instructions coded in the SCNAME column are valid. To specifically debug such mapping instructions, there is a special debugging flag you can enable within the SCAMO.JS script code. Near the beginning of the script, look for the line which reads:

  ```
  debugMapping = 0;
  and change it to read
  debugMapping = 1;
  ```

  Then restart the SCAuto for AMO service (or manually rerun the scamo.js script from the command line or by double-clicking it in Explorer) and consult the scamo.log file. It should contain detailed debugging messages like these to help you debug your mapping instructions:
- There is also a debugging statement that can be enabled in the *scamo.js* script. Near the beginning of the script, look for:

  ```
  debugMapping = 0;
  ```

  and change the assignment from 0 to 1. Then restart the script and check the *scamo.log* file. It should contain messages like these to help you debug your mapping instructions:

  ```
  07/21/1999 15:25:00   pid (309)(334) Processing AMO data
  item 5 2 9 $GeneralInventory$ $Display Adapter$ BIOS info
  : "ATI Graphics Accelerator"
  07/21/1999 15:25:00   pid (309)(334) Mapping instructions
  are: "video"
  07/21/1999 15:25:00   pid (309)(334) Appending "ATI
  Graphics Accelerator" to the end of scalar field "video"
  07/21/1999 15:25:00   pid (309)(334) Processing AMO data
  item 5 2 18 $GeneralInventory$ $Display Adapter$ Color
  depth : "True Color"
  07/21/1999 15:25:00   pid (309)(334) Mapping instructions
  are: "video"
  07/21/1999 15:25:00   pid (309)(334) Appending "True
  Color" to the end of scalar field "video"
  07/21/1999 15:25:00   pid (309)(334) Processing AMO data
  item 5 2 104 $GeneralInventory$ $Display Adapter$ Name :
  "ati"
  07/21/1999 15:25:00   pid (309)(334) Mapping instructions
  are: "video"
  07/21/1999 15:25:01   pid (309)(334) Appending "ati" to
  the end of scalar field "video"
  ```

```
07/21/1999 15:25:01   pid (309)(334) Processing AMO data
item 5 2 144 $GeneralInventory$ $Display Adapter$
Resolution : "800,600"
07/21/1999 15:25:01   pid (309)(334) Mapping instructions
are: "video; feature.size[ 16 ]; feature.id[16] = 16;
feature.description[16] = ITNAME3;"
07/21/1999 15:25:01   pid (309)(334) Appending "800,600"
to the end of scalar field "video"
07/21/1999 15:25:01   pid (309)(334) Setting
feature.size array element  16  to 800,600
07/21/1999 15:25:01   pid (309)(334) Setting  feature.id
array element 16 to 16
07/21/1999 15:25:01   pid (309)(334) Setting
feature.description array element 16 to Resolution
07/21/1999 15:25:01   pid (309)(334) Processing AMO data
item 5 2 186 $GeneralInventory$ $Display Adapter$ Type :
"ATI 3D RAGE IIC (GT-B3U1)"
07/21/1999 15:25:01   pid (309)(334) Mapping instructions
are: "video"
07/21/1999 15:25:01   pid (309)(334) Appending "ATI 3D
RAGE IIC (GT-B3U1)" to the end of scalar field "video"
```

After you have corrected your mapping instructions, reassign the debugging parameter to 0:

```
debugMapping = 0;
```

to disable debugging and improve performance.

- If events are being generated, but are not being processed correctly in ServiceCenter once they arrive, check the following:

  Make sure you have the new ICM event types defined in your ServiceCenter Event Services. If you are using a supported version of ServiceCenter (2.1 SP 3E2 or later, or 3.0 or later) for this product, and if you applied the required unload files specified in the README.TXT, then the new ICM events including *ICMpc* and *ICMserver* should exist. Go into Event Services in ServiceCenter and verify that these event definitions exist. They should contain all of the fields in the device and associated attribute file, in the same order as the DBDICT, starting with *logical.name*.

Chapter 6 Scripting

Overview

This chapter explains how Microsoft JScript 5.0 is used in the SCAuto for Unicenter AMO gateway, and how other ActiveX scripting languages such as VBScript 5.0 may be used in addition to, or instead of JScript. It also discusses some of the logic of the provided script called scamo.js, which is the main program of the gateway.

Windows scripting architecture and terminology

Until recently, the only native scripting language supported by the Windows operating system was the old MS-DOS® operating system command language used for batch files (.bat). This was a serious limitation of Windows. With the new Windows scripting architecture, users can take advantage of powerful scripting languages such as Visual Basic Scripting Edition and JScript. These scripting languages support object-oriented programming using OLE Automation objects (COM).

The Windows scripting architecture has three components:

ActiveX scripting hosts

These are programs such as the Microsoft-provided WSCRIPT.EXE and CSCRIPT.EXE. A scripting host is a program that can load an ActiveX scripting engine and a text file containing a script, and *feed* the script to the engine. A scripting host may also implement various objects and make these available to the script programmer.

ActiveX scripting engines

A scripting engine is a .DLL file which implements a particular scripting language. Microsoft currently provides two different scripting engines, one for JScript and another for VBScript. These are included in the Windows Script 5.0 distribution described in *Chapter 2*.

scripts written in one of the supported scripting languages

Windows scripting is done by writing a script in a supported language and invoking a scripting host such as wscript.exe to process the script. The script can use any OLE Automation objects that have been installed on the Windows platform where the script is running. You may be surprised at the number of such objects that can be found on any given Windows system.

Microsoft scripting engine considerations

The SCAuto for Unicenter AMO scripts use Microsoft JScript 5.0, which is an ECMA262 compliant implementation of ECMA script.

Because it makes use of Microsoft extensions not yet in the ECMA specification, such as exception handling using *try...catch* notation, it is important to be sure that Microsoft JScript version 5.0 or later is installed on the NT platform where SCAuto AMO is running. Versions of JScript prior to 5.0 do not contain support for exception handling.

Although we chose to use JScript to implement SCAuto for Unicenter AMO, the OLE Automation objects provided by Peregrine can be used just as easily from VBscript as from JScript. So if you have a major project to tackle, such as developing an additional script to integrate some other Unicenter AMO application with ServiceCenter, you can do it in VBScript if you prefer.

Windows scripting host considerations

No particular use is made of the special *wsh* objects or other Windows Scripting Host specific functionality in this release. wscript.exe is used to invoke and run the scamo.js script. The command shell cscript.exe could just as easily be used instead.

The first release of SCAuto for Unicenter AMO uses Windows Scripting Host 1.0, because WSH 2.0 was not yet generally available.

If you plan to do significant customization, you should investigate Windows Scripting Host 2.0, which supports multiple scripting languages in the same file, inclusion of scripts from other files, and other procedures.

Recommended tools

There are a variety of tools available for free download from the Microsoft scripting web site referenced earlier in this book. These include a simple script debugger. However, it is recommended that you instead install the much more capable script debugger included in the Visual Studio Professional or Enterprise edition product.

No script customization should be attempted without first installing some sort of script debugger. If you don’t have a version of Visual Studio which includes the enhanced script debugger, download the free script debugger from the Microsoft scripting Web site.

The SCAMO.JS Script

This script functions as a main program of the SCAuto AMO gateway. It collects information about hardware devices, maps it to ServiceCenter fields, and writes it to an external file. It is executed by the command wscript.exe scamo.js, which appears in the *scamo.cfg* file. The latter file is processed by the SCAuto AMO service control program, scamosrv.exe, at startup; and commands in the file are executed. You can make the gateway execute additional scripts by adding additional lines to the scamo.cfg file to invoke wscript.exe for the new script.

The scamo.js script is essentially an endless loop that runs until the SCAutoAMO service is shut down, which causes scamosrv.exe to terminate the wscript process(es) that it started at startup. After some initialization code is executed to read parameters from scamo.ini and create some OLE Automation objects, the script executes continuously until service shutdown.

If the SCAutomate services are running, the events written to *scevents.to.<host.port>* will be sent to the ServiceCenter *eventin* queue, where they will be processed by Event Services and inserted into ServiceCenter's Inventory repository.

Figure 6-1 provides an overview of the scamo.js script.

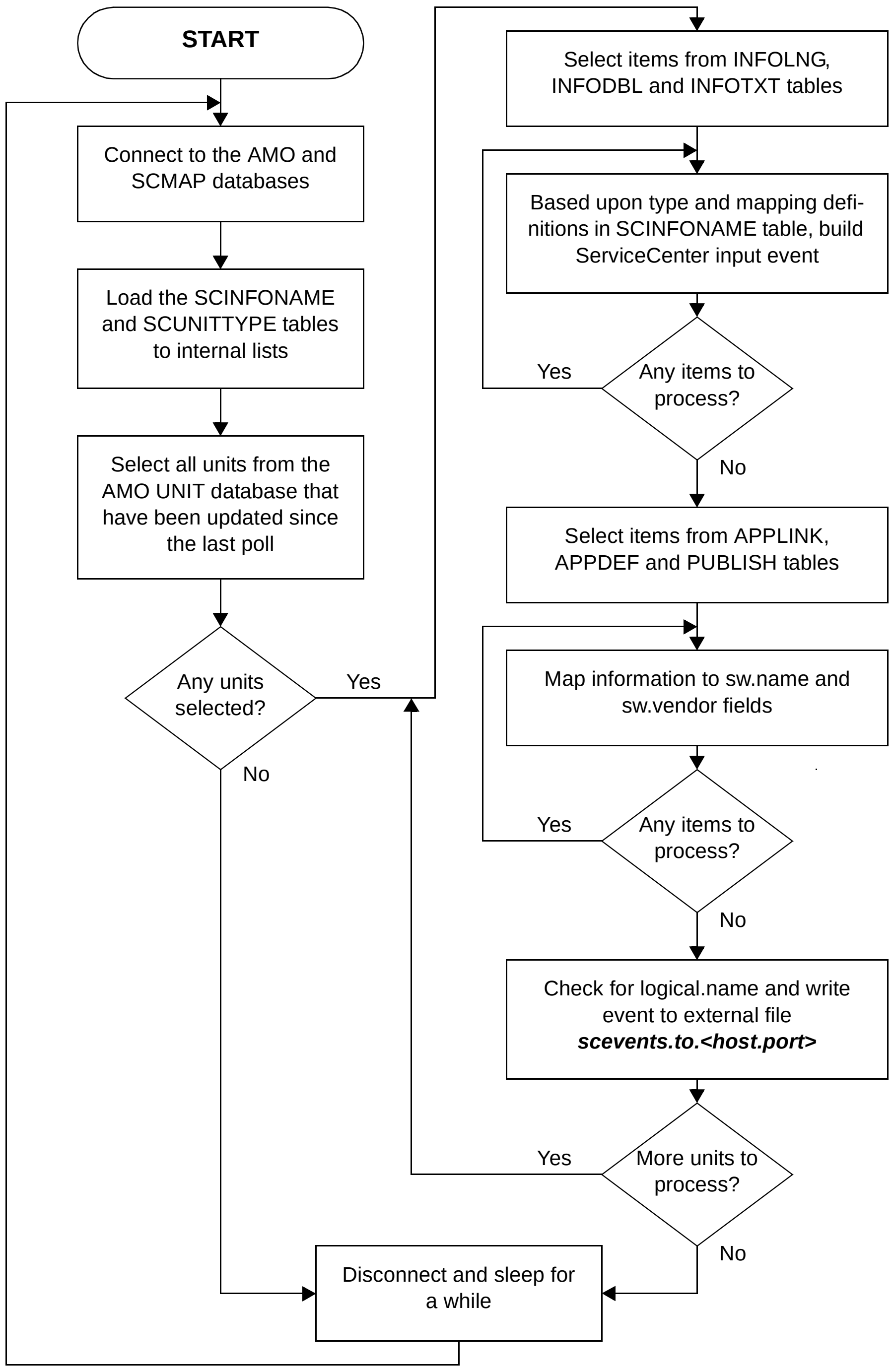

START
Connect to the AMO and SCMAP databases
Load the SCINFONAME and SCUNITTYPE tables to internal lists
Select all units from the AMO UNIT database that have been updated since the last poll
Any units selected?
Yes
No
Select items from INFOLNG, INFODBL and INFOTXT tables
Based upon type and mapping definitions in SCINFONAME table, build ServiceCenter input event
Any items to process?
Yes
No
Select items from APPLINK, APPDEF and PUBLISH tables
Map information to sw.name and sw.vendor fields
Any items to process?
Yes
No
Check for logical.name and write event to external file ***scevents.to.<host.port>***
More units to process?
Yes
No
Disconnect and sleep for a while

Figure 6-1 scamo.js Script

The MAKESCINFO.JS Script

The MAKESCINFO script builds the SCINFONAME table based upon parameters defined in the sc.ini file. The purpose of the SCINFONAME table is to define mapping instructions such that attributes detected by Unicenter AMO can be written to ServiceCenter's Inventory tables.

SCAuto for Unicenter AMO is delivered with an SCINFONAME table that contains entries for PC and Server inventory items. These items are defined as having a unit TYPE of 1 (Computer), 100 (NetWare Server), 110 (Vines Server) and 120 (NT Server), as specified in the *unittypes* parameter of the sc.ini file. Each item is assigned a DOMAINID of 0, since the default assumption is that each item of the same type has the same attributes across all domains (as defined by the *domainid* parameter in the sc.ini file). Excluded are components described by the ITRTNMID setting that have no corresponding column in the ServiceCenter Inventory tables, as defined in the sc.ini file's *excludeitems* parameter.

Usually, unless inventory attributes differ across domains there is no need to rebuild the SCINFONAME table. However, if your specifications require enlarging the table, the makescinfo.js script is the tool for doing so.

Figure 6-2 shows how the MAKESCINFO.JS script rebuilds the SCINFONAME table. If you decide you need to run MAKESCINFO.JS, do the following:

- Shut down the SCAuto AMO service.
- Make a backup copy of the *\Data\scamo.mdb* file.
- Using Microsoft Access:
  - Open *\Data\scamo.mdb*.
  - Open the SCINFONAME table.
  - Delete all the records in the SCINFONAME table.
  - Cose *\Data\scamo.mdb*.
- Modify the *excludeitems* and *domainid* settings in *scamo.ini* (only if strictly necessary).
- From the command line, run *\Bin\makescinfo.js*.
- Monitor the *scamo.log* file for any errors.

- When the script completes, using Microsoft Access:
  - Open *Data\scamo.mdb*
  - Open the SCINFONAME table
  - Enter ICM field names and other mapping instructions in the SCNAME column for each AMO data item you want mapped. It is recommended that you examine your backup copy of *scamo.mdb*, or a printout of it, to help you do this correctly
  - close *Data\scamo.mdb*.

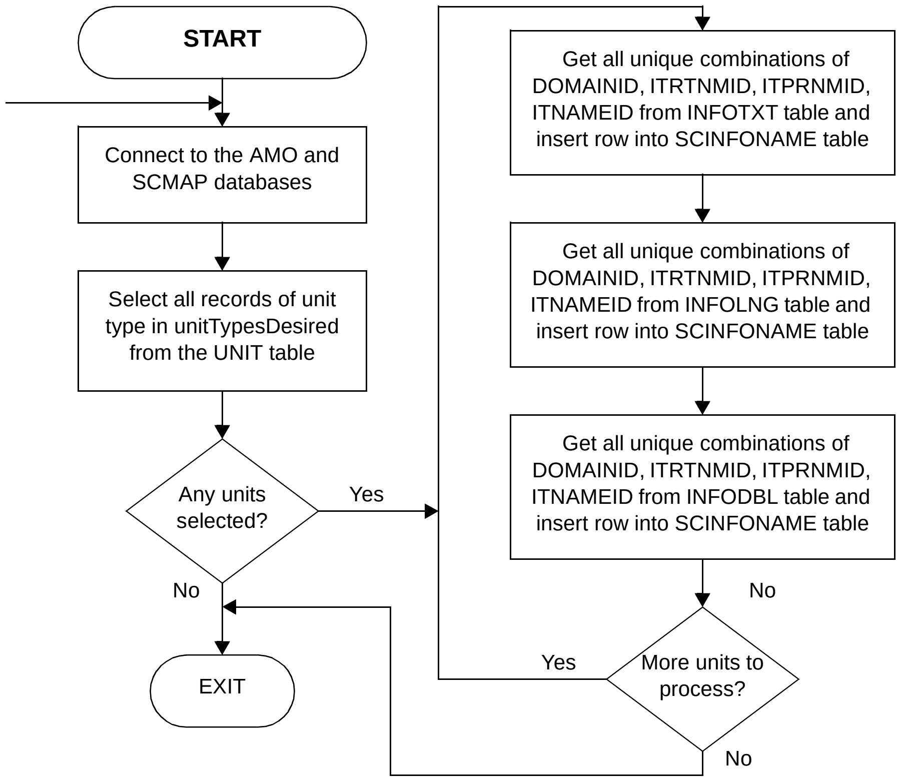

START
Connect to the AMO and SCMAP databases
Select all records of unit type in unitTypesDesired from the UNIT table
Any units selected?
Yes
No
EXIT
Get all unique combinations of DOMAINID, ITRTNMID, ITPRNMID, ITNAMEID from INFOTXT table and insert row into SCINFONAME table
Get all unique combinations of DOMAINID, ITRTNMID, ITPRNMID, ITNAMEID from INFOLNG table and insert row into SCINFONAME table
Get all unique combinations of DOMAINID, ITRTNMID, ITPRNMID, ITNAMEID from INFODBL table and insert row into SCINFONAME table
No
More units to process?
Yes
No

Figure 6-2 The MAKESCINFO.JS Script

Once the SCINFONAME table has been rebuilt, you must edit it to redefine mapping instructions for any new rows.

Chapter 7 Operating SCAuto for Unicenter AMO

Overview

This chapter provides the procedures for operating SCAuto for Unicenter AMO.

Starting SCAuto for Unicenter AMO

There are three ways to start SCAuto for Unicenter AMO:

- Choose **Start->Programs->SCAuto for Unicenter AMO->Start SCAuto for Unicenter AMO** from the menu.

  This executes the command **scamosrv -start** from the **\SCAuto for Unicenter AMO\Bin** directory.
- Issue the following command from a Command window (the current directory in the command window is unimportant):

  net start scautoamo
- Execute the Control panel services applet to start SCAuto for Unicenter AMO.

All three of these methods execute the same code, and are completely identical in function.

Starting the **SCAUTOamo** service normally takes from two to five seconds. A normal startup of the service generates messages in the *scamo.log* file like this:

```
07/02/1999 12:27:20   pid (435)(461) =================
SCAuto for Unicenter AMO started ==================
07/02/1999 12:27:20   pid (435)(461) scamosrv.exe
(Linked: Thu Jul 01 12:06:21 1999) started
07/02/1999 12:27:20   pid (435)(461) scamosrv.exe:
Created stop event SCAutoAMO.StopEvent
07/02/1999 12:27:20   pid (435)(461) Changing desktop for
process wscript.exe from Default to winsta0\default
```

```
07/02/1999 12:27:20   pid (498)(457) scevmon.exe (Linked:
Thu Jul 01 12:04:57 1999) started
07/02/1999 12:27:20   pid (498)(457) scevmon: Using stop
event SCAutoAMO.StopEvent
07/02/1999 12:27:20   pid (498)(457) scevmon.exe:
scauto_init: SCAUTO.DLL version is SCAuto SDK 3.2.0,
built Jul  1 1999 12:02:33
07/02/1999 12:27:20   pid (498)(457) scevmon:
"noeventsfromsc" parm recognized. No events will be
retrieved from ServiceCenter EventOut queue
07/02/1999 12:27:20   pid (337)(496) ==============
SCAuto for Unicenter AMO: Script scamo.js started
=============
07/02/1999 12:27:20   pid (337)(496) Home directory is
D:\PROGRA~1\PEREGR~1\SCAUTO~4
07/02/1999 12:27:20   pid (337)(496) Using .ini file
D:\PROGRA~1\PEREGR~1\SCAUTO~4\scamo.ini
07/02/1999 12:27:20   pid (337)(496) Successfully created
the SCEvent object
07/02/1999 12:27:20   pid (337)(496) wscript.exe:
scauto_init: SCAUTO.DLL version is SCAuto SDK 3.2.0,
built Jul  1 1999 12:02:33
07/02/1999 12:27:20   pid (337)(496) Using AMO database
connection string Driver={Microsoft Access Driver
(*.mdb)}; DBQ=D:\AVNET\DB1.MDB; DefaultDir=D:\AVNET\
07/02/1999 12:27:20   pid (337)(496) Using SC Mapping
Database connection string Driver={Microsoft Access
Driver (*.mdb)}; DBQ=SCAMO.MDB;
DefaultDir=D:\PROGRA~1\PEREGR~1\SCAUTO~4\Data
07/02/1999 12:27:20   pid (498)(457) scevmon : Connected
to localhost.12690
07/02/1999 12:27:20   pid (337)(496) Successfully created
the ADO database objects
07/02/1999 12:27:21   pid (337)(496) Successfully
connected to the AMO database
07/02/1999 12:27:21   pid (337)(496) Successfully
connected to the SCAMO mapping database
```

If the service is unable to start for any reason, a message will be logged in the Windows NT system log, and detailed error messages will appear in the *scamo.log* file.

What happens when SCAuto for Unicenter AMO is started

When SCAuto for Unicenter AMO is started, the main service executable ***scamosrv.exe*** is invoked. It then locates the file *scamo.cfg*, which is found in the **\SCAuto for Unicenter AMO** directory. If this file cannot be found, the service will not start.

Next, the *scamo.cfg* file is processed. Each line in the file that is not blank or commented is interpreted as a process to run. Processes are typically either transient commands (i.e., invocations of *cmd.exe*) or one of the following background processes installed with SCAuto for Unicenter AMO:

scevmon.exe

This process makes a TCP connection with the SCAUTOD component of ServiceCenter, using the hostname and port number information supplied to the configuration dialog program *scamocfg.exe*. Whenever a connection is established, scevmon attempts to exchange events with ServiceCenter. **scevmon.exe** manages two local queues of events: an inbound events queue and an outbound events queue. These queues are contained in the SCAuto for Unicenter AMO installation directory.

The name of the outbound queue is **scevents.to.<host>.<port number>**. Outgoing events are logged to the *scevents.to.* file by processes such as the transient message action command **scevent.exe** or the long-running background process and consumed by scevmon.exe, which sends them to the ServiceCenter SCAUTOD process indicated in the file name of the **scevents.to** queue. When SCAUTOD accepts the events, they are written to the ServiceCenter EVENTIN file.

The name of the inbound events queue is **scevents.from.<host>.<port number>**. This queue is created by **scevmon.exe** from events retrieved from the ServiceCenter EVENTOUT file via SCAUTOD. **scevmon.exe** determines its starting position in the EVENTOUT file of ServiceCenter by consulting a file called *syncfile.<host>.<service>* where *<host>* and *<service>* are the hostname and TCP port or service name of the SCAUTOD server. It then requests from SCAUTOD all events in EVENTOUT that have sequence numbers greater than the value contained in the *syncfile*. The *syncfile* is updated as each event is received from SCAUTOD and successfully written to the *scevents.from* queue.

After startup, SCAuto for Unicenter AMO generates one ICM event for each AMO *unit* (a term similar to the SMS term *machine*). For hardware, some of the AMO units are personal computers, some are Microsoft NT servers and some are Novell servers. Although there are numerous other types, the three aforementioned types are the only ones processed without customization. To customize, update the SCUNITTYPE table and the list of unit types found in the *scamo.ini* file.

SCAuto for Unicenter AMO decides what information to put in each sub field of the ICM events by consulting the **SCINFONAME** table. Upon startup, without doing any customization, you should see ICM events being generated. Look in the scamo.log file for a number of messages such as these:

```
08/09/1999 09:42:16   pid (136)(246) ===================
SCAuto for Unicenter AMO started ===================
08/09/1999 09:42:16   pid (136)(246) scamosrv.exe
(Linked: Thu Jul 01 12:06:21 1999) started
08/09/1999 09:42:16   pid (136)(246) scamosrv.exe:
Created stop event SCAutoAMO.StopEvent
08/09/1999 09:42:16   pid (136)(246) Changing desktop for
process wscript.exe from Default to winsta0\default
08/09/1999 09:42:16   pid (262)(254) scevmon.exe (Linked:
Thu Jul 01 12:04:57 1999) started
08/09/1999 09:42:16   pid (262)(254) scevmon: Using stop
event SCAutoAMO.StopEvent
08/09/1999 09:42:16   pid (262)(254) scevmon:
"noeventsfromsc" parm recognized. No events will be
retrieved from ServiceCenter EventOut queue
08/09/1999 09:42:18   pid (268)(239) ===================
SCAuto for Unicenter AMO: Script scamo.js started
===================
08/09/1999 09:42:18   pid (268)(239) Home directory is
c:\SCAUTO~1
08/09/1999 09:42:18   pid (268)(239) Using .ini file
c:\SCAUTO~1\scamo.ini
08/09/1999 09:42:18   pid (268)(239) Successfully created
the SCEvent object
08/09/1999 09:42:18   pid (268)(239) Using AMO database
connection string Driver={Microsoft Access Driver
(*.mdb)}; DBQ=..\data\db1; DefaultDir=..\data\
08/09/1999 09:42:18   pid (268)(239) Using SC Mapping
Database connection string Driver={Microsoft Access
Driver (*.mdb)}; DBQ=SCAMO.MDB;
DefaultDir=c:\SCAUTO~1\Data
08/09/1999 09:42:18   pid (268)(239) Successfully created
the ADO database objects
```

```
08/09/1999 09:42:20   pid (268)(239) Successfully
connected to the AMO database
08/09/1999 09:42:20   pid (268)(239) Successfully
connected to the SCAMO mapping database
08/09/1999 09:42:26   pid (268)(239) Successfully logged
event MA1951-D2-ORACLE^^^^^^^^^4.10 NetWare 4.10^November
8,
1994^^Novell^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:28   pid (268)(239) Successfully logged
event UFSNW0001CORP^^^^^^^^^4.11 NetWare 4.11^August 22,
1996^^Novell^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:30   pid (268)(239) Successfully logged
event S1-165-01^^^^^^^^^4.10 NetWare 4.10^November 8,
1994^^Novell^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:31   pid (268)(239) Successfully logged
event NY1101-M1-ISDNY^^^^^^^^^4.10 NetWare 4.10^November
8,
1994^^Novell^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:34   pid (268)(239) Successfully logged
event MA1951-M3-H218^^^^^^^^^4.10 NetWare 4.10^November
8,
1994^^Novell^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:38   pid (268)(239) Successfully logged
event global center^Example: (Last Name, First
Name)^^^^^^^^^^pc^^^172.16.23.38 172.16.23.38^^h07705f
h07705f^^^^^^^^^^172.16.23.201^255.255.255.0
255.255.255.0^avnet^^^03/07/97
Phoenix^^^^^h07705f^^^^^00AA00386B71^^^^^^^^P5 Intel
Pentium P54C (3.3V) P5 Intel Corporation^Super VGA
800x600 (Standard monitor types)^Bits Per Pixel=16
Hewlett Packard VL5 S3 Trio 64V2 800,600 HP/S3^ethernet
Intel EtherExpress 16 or 16TP Intel^^^^^^^^^^
08/09/1999 09:42:42   pid (268)(239) Successfully logged
event U004WKNT0016^Example: (Last Name, First
Name)^^^^^^^^^^pc^^^172.16.23.5^^u004wknt0016^^^^^^^^^^17
2.16.23.201^255.255.255.0^avnet.com^^^11/10/97
PhoenixBIOS 4.0 Release
6.0^^^^^H097T92^^^^^00A0C98CDAF4^^^^^^^^P6 Intel
PentiumPro P6 P6 Intel Corporation^^NV3 True Color
stbv128m 1024,768 NV3^Ethernet [1] Intel 82557-based 10/
100 Ethernet PCI Adapter Intel^^^^^^^^^^
```

The rate at which the ICM events are generated depends on several factors:

- The number of milliseconds of wait time introduced into the script. If you have a fast server, or if the database is in SQL Server on another machine, you should set the wait interval to zero. Locate the line of code toward the top of the *\Bin\scamo.js* or *\Bin\scamosw.js* file that says:

  ```
  perItemSleepInterval = 1;
  ```

  and change it to set the value to 0, instead of 1.
- The CPU speed and number of processors.
- The network speed and the speed of the database machine, if the database is on another machine.

Under ideal conditions, SCAuto for Unicenter AMO should be able to process about four AMO units per second. The initial loading may take several hours in a large shop with 10,000 or more units in the database.

After processing all the units, the product will wait a user-defined interval before querying AMO to see if any of the units have been updated. This is done by selecting units with an LRUNDATE date value larger than that previously seen for any of the units previously processed.

Note: This wait interval parameter is defined in the *scamo.ini* file, and is specified as *sleep:3600* (which is one hour) by default.

After the initial run, updates should arrive at the AMO database from time to time, rather than all at once.This is dependent upon the way you configure AMO. The performance of SCAuto for Unicenter AMO is far better if the inventory collections done by AMO are periodic (during each individual PC reboot or login, or at least all machines in one domain at a time) rather regularly scheduled.

TOC PREV NEXT INDEX

Stopping SCAuto for Unicenter AMO

There are three ways to stop SCAuto for Unicenter AMO:

- Choose **Start/Programs/SCAuto for Unicenter AMO/Stop SCAuto for Unicenter AM*O*** from the menu.

  This executes the command **scamosrv -stop** from the **\SCAuto for Unicenter AMO\Bin** directory.
- Issue the following command from a Command window. The current directory in the command window is unimportant:

  net stop scautoamo
- Execute the **Control Panel Services** applet to stop SCAuto for Unicenter AMO.

All three of these methods execute the same code, and are identical in function.

Stopping the SCAUTOamo service normally takes 5 to 10 seconds, except when CPU monitoring is in effect. If the service was instructed to monitor the CPU utilization via the optional monitorcpu:1 parameter in the *scamo.ini* file, stopping the SCAutoamo service may take up to 20 seconds.

What happens when SCAuto for Unicenter AMO is stopped

Stopping the SCAutoAMO service signals a Windows NT event called SCAutoamo.StopEvent. This causes the various processes that were started by the service to immediately go into termination logic, where they release whatever resources they acquired from Windows NT and then exit. You can verify that the service stopped normally by checking the scamo.log file, where you should see a series of messages like these:

```
08/09/1999 09:42:35   pid (262)(254) scevmon: Stop event
recognized. Terminating.
08/09/1999 09:42:35   pid (136)(246) Stop event
recognized; waiting for child processes to terminate
08/09/1999 09:42:35   pid (268)(239) Successfully logged
event SFT3^^^^^^^^^^3.11 NetWare SFT III MSEngine - v3.11
B (10 user)^1/8/
93^^Novell^^^^^^^^^^server^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^
08/09/1999 09:42:35   pid (268)(239) Execution
interrupted by service shutdown
08/09/1999 09:42:35  pid (268)(239) Waiting 3600 seconds
till next AMO Database poll
```

```
08/09/1999 09:42:35   pid (268)(239) Shutdown event
signalled; terminating..
08/09/1999 09:42:35   pid (268)(239) ===================
SCAuto for Unicenter AMO: Script scamo.js completed
===================
08/09/1999 09:42:36   pid (136)(246) All processes
terminated
08/09/1999 09:42:36   pid (136)(246) ===================
SCAuto for Unicenter AMO shutdown complete
===================
```

Should any process fail to exit voluntarily for any reason, the main service executable scamosrv.exe will terminate it using the Windows NT **Terminate Process** command. For this reason, the service should not normally fail to stop. If the service fails to stop, detailed error messages will appear in the *scamo.log* file.

Determining if Unicenter AMO is Running Normally

You can use the Windows NT Resource Kit command **SC** (Service Control) to query the SCAUTOamo service as follows:

sc query scautoamo

This results in output similar to the following:

```
N:\tmp>sc query scautoamo
SERVICE_NAME: scautoamo
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
```

An alternative way to determine if SCAuto for Unicenter AMO is running is to use the Task Manager. If the service is running, the **scamosrv.exe** process is visible in the task list displayed by Task Manager.

Day to Day Administration

SCAuto for Unicenter AMO is designed for minimal administration overhead and maximum availability and operational flexibility.

SCAuto for Unicenter AMO can tolerate indefinite outages of Unicenter AMO and ServiceCenter. The operation of the latter products does not have to be synchronized with SCAuto for Unicenter AMO (i.e., SCAuto for Unicenter AMO does not have to be stopped if ServiceCenter is being stopped or paused, nor is it necessary for those products to be running in order to start SCAuto for Unicenter AMO).

SCAuto for Unicenter AMO does not maintain a database. It produces a log file, but it automatically wraps this log when it reaches a defined maximum size. The default maximum log file size is 5 MB. This value is controlled by a setting in the *scamo.ini* file. You should never have to stop SCAuto for Unicenter AMO.

SCAuto for Unicenter AMO can be started and stopped at any time. However, long periods of Unicenter AMO operation without running SCAuto for Unicenter AMO should be avoided, as this will result in out-of-date information in the ServiceCenter database.

Similarly, long periods of Unicenter AMO and SCAuto for Unicenter AMO operation without running ServiceCenter should be avoided, as large numbers of events may accumulate in the *scevents.to...* file awaiting transmission to ServiceCenter.

TOC PREV NEXT INDEX

Index

TOC PREV NEXT INDEX

Peregrine SYSTEMS®

The Infrastructure Management Company™